LLNL-TR-637714

LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

Tracer Particle Locations with Curved or Arbitrary Planar Meshes

J. Yao

May 29, 2013

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Tracer Particle Locations with Curved or Arbitrary Planar Meshes

Yao, Jin

Lawrence Livermore National Laboratory, California, USA

Abstract

Determining distributed particle locations in a mesh of high-order elements is challenging in the sense that not only a full inclusion test is cost inefficient, but also the inclusion test with an element with curved boundary is not trivial as in the case of a planar polyhedron. An algorithm to identify tracer particle locations in a high-order element system (or an arbitrary planar mesh) is proposed. The algorithm does only inclusion tests for particles near their owner elements thus is complete local. This feature would be beneficial for parallelization of the method.

1 INTRODUCTION

Testing the locations of tracer particles with a high-order element system has practical use for simulation of important physical processes in which complex geometries present. Unlike inclusion tests for elements with planar or linear boundaries, a face of a high-order element can be difficult to handle algebraically. Furthermore, an effective method to solve this problem has to be a local one, otherwise the cost associated with solving high order algebraic equations shall make a global method impractical, not to mention that parallelization of a global test is not acceptable for exascale computations.

In this report, a local searching method to locate tracer particles in a high-order element system is proposed. The method finds for each particle the elements which are logically close, and does a simplified inclusion test to determine which element contains a given particle.

A background virtual cubical complex is employed to cover the computational domain. Finding elements that intersect a non-empty virtual cube is the key issue to the solution of the problem. A global method using pseudo nodes inside an element, or a local method utilizing directional walks[1] on the element boundary that enters

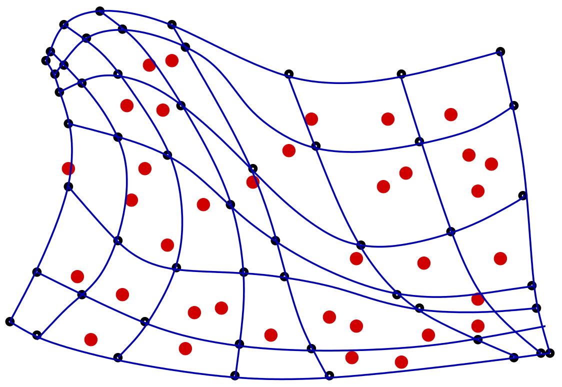

Figure 1: *A two-dimensional mesh with curved cell faces with a random distribution of tracer particles.*

and exits from virtual cubes effectively determine for each non-empty virtual cube (means the cube contains some tracer particle(s)) a list of elements that intersects the cube. Then for each tracer particle, simplified local inclusion tests against the elements that intersect the owner cube of the particle are conducted to accurately locate the element that contains the particle.

The solution approach in three-dimensions is a direct extension of the two-dimensional solution method proposed here, thus only the two-dimensional solution is demonstrated with figures in this article. Similar approaches would work for an arbitrary planar mesh.

This local method is easy to parallelize and should fit X-scale calculations well.

2 THE ALGORITHM

2.1 DEFINITION OF THE PROBLEM

A set of N particles is randomly distributed in space (two-dimensional or three-dimensional) and contained in a physical mesh with high-order elements. The question is how to find for each particle the element that contains the particle (fig. 1).

As a necessary pre-requirement, an effective point inclusion test with a high-order element must be provided and in this report such an inclusion test will be described.

The solution approach in three-dimensions is a direct extension of the two-dimensional solution method proposed here. Only the two-dimensional solution is described in this article.

2.2 THE SCALING OF THE PROBLEM

We choose a special length ℓ_c to scale the physical system. ℓ_c probably should be comparable to the average element dimension. It may be selected to be the square root (cubic root in 3D) of the average volume of elements. Furthermore for convenience the origin can be set at $(x_0, y_0) = (x_{Min}, y_{Min})$ where x_{Min} and y_{Min} are the lower bounds of the coordinates of the nodes of elements. After that we scale the problem with ℓ_c thus the system is nondimensionalized.

2.3 ORIENTATION OF AN ELEMENT

A curved element in two-dimensions is assumed to have its nodes ordered counter-clock-wise, therefore with the right hand rule, a directional walk on the boundary of an element keeps the interior of the element on the left (this is to say, if one raises his arms parallel to the plane the element resides in, his right arm will be in the boundary normal direction pointing exterior).

A curved element in three-dimensions is assumed with the nodes on each element face also ordered counter-clock wise.

The orientation of an element described above is essential for a successful directional walk[1] to find intersections of walls of a virtual cube and element boundaries, in order to obtain locations of the tracer particles.

2.4 THE VIRTUAL CUBICAL COMPLEX

Assume the problem has been scaled by ℓ_c in space, with each element obeying the desired orientation described in the last section.

The entire computational therefore is contained in a Cartesian grid, or a virtual cubical complex (fig. 2). Imagine the unit square defined by diagonals $(i, j), (i + 1, j + 1)$ be virtual cubical (i, j), each of the given point $p = (x, y)$ must be contained in such a cube. The cube ID that owns p is simply the integer parts of x and y (fig. 3).

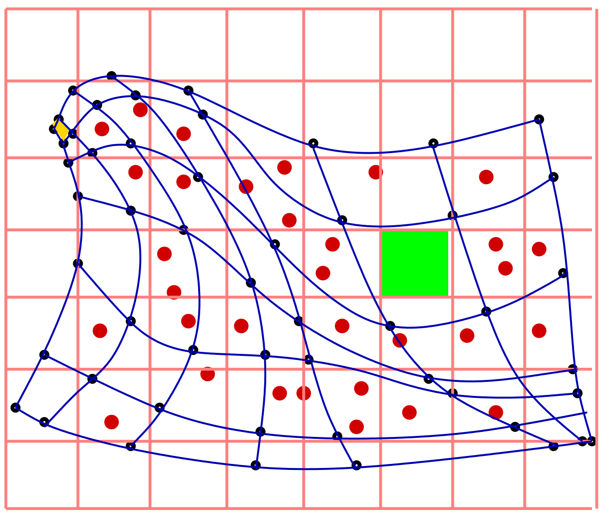

Figure 2: *A virtual cubical complex that covers the curved mesh. Note the yellow element is completely contained in a cube, while the green cube is completely contained in an element.*

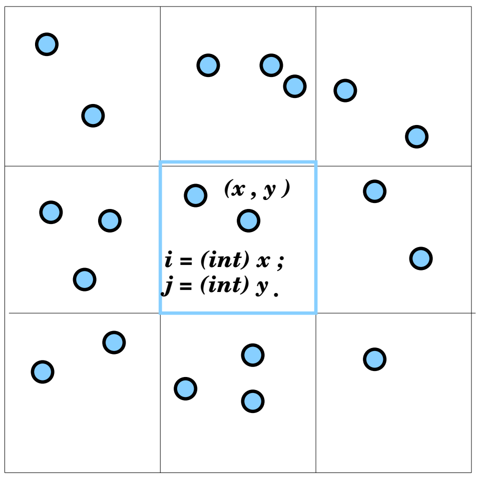

Figure 3: *The cube ID that contains a given point is simply the integer parts of the physical coordinates of the point.*

2.5 DECOMPOSE THE PROBLEM TO A COLLECTION OF SOLUTIONS IN A VIRTUAL CUBE

The search of tracer particle locations in a curved element system can be reduced to a collection of local problems of the same nature with several steps.

1) scale the system with a characteristic length;

2) build a virtual cubical complex with a Cartesian integer grid to cover the computational domain; and

3) find for each tracer particle the virtual cube contains the particle, also find for each element node its virtual cube owner.

From now on, the whole problem is converted to finding the elements that overlap the virtual cube that contains some particles, then performing local inclusion tests for a particle.

We have two ways to deal with the problem starting from here, the first is easy to implement, does not require a walk on element boundary to find intersections with a virtual cube. However it is a global method. The second approach is a local one and gives exact solution, but requires a directional walk to find intersections of an element and a virtual cube.

2.5.1 DISTRIBUTION OF PSEUDO NODES

Assume an element is mapped from its logical space. If the element is normal, an interior point in logical space shall be mapped to the interior in physical space. One can distribute points interior to the element in the logical space and map them into physical space. A choice maybe to divide the logical element (a 2 by 2 square in logical space, say) to k by k squares of the same size and map the nodes of these squares in to the physical space (they might be called "pseudo nodes", see fig. 4), make sure every sub-element has its maximum dimension smaller than the dimension of a virtual cube (this can be done by evaluating the maximum value of local Jacobian of the mapping).

After every element has their pseudo nodes mapped into the physical space, a double loop, first over the elements, then for each element, over its pseudo nodes, shall tell the ID of virtual cube that contains each pseudo node. The cube ID is simply the integer parts of the physical coordinates of a pseudo node. If a cube that contains some tracer particles also contains any pseudo node from a certain element,

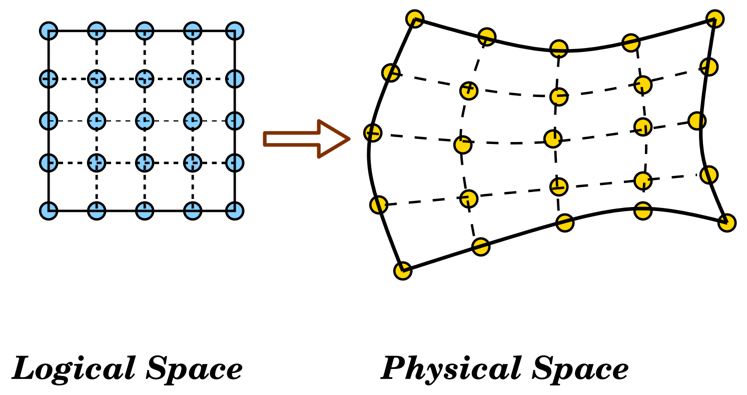

Figure 4: *Pseudo nodes mapped from logical space into physical space.*

collect this virtual cube and add the element to the list of elements that intersect that cube.

Then for each virtual cube that contain some tracer particles, the elements that intersect the cube are identified. However this list of local elements may be incomplete because the image (a sub-element) of a sub-square in the logical space, may intersect a virtual cube but with none of its pseudo node inside the cube, for a boundary sub-element (fig. 6).

Since the maximum dimension of a sub-element is smaller than the size of a virtual cube, if a virtual cube is interesting any element, there must be at least one pseudo-node of some element falls into this cube. The we have identified an element that intersects the cube. If this cube also contains a tracer particle, one shall perform a local inclusion test for this particle to each element that has a pseudo-node contained in the virtual cube. Once a particle is found to have no local element containing it, one looks for elements that are neighbor (sharing a cell face) to the known local elements and do local inclusion tests with the neighbor elements until an element is found to contain the particle. The operation in the next section can also be performed once for the same purpose.

The above discussion is for a global method for this problem. A local logic to find the exact solution is described in the next section.

2.5.2 LOOP OVER LEVEL OF NEIGHBORS

The exact solution can be obtained with the following steps.

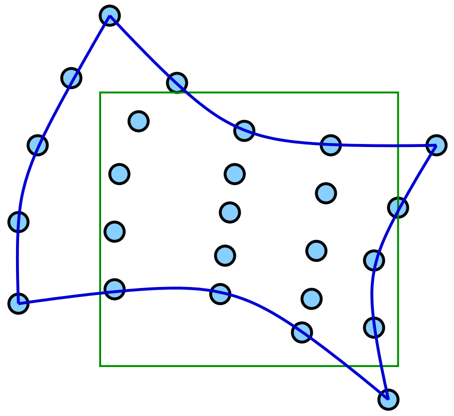

Figure 5: *An element with no corner nodes contained in the virtual cube (green square), but pseudo nodes helped to find it intersects the cube.*

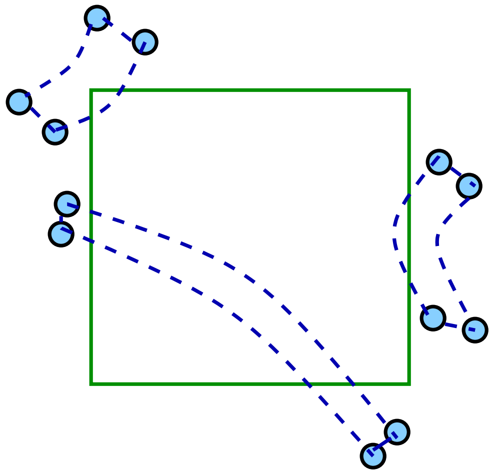

Figure 6: *Some possibilities that none of the pseudo nodes is contained in a virtual cube, but the element intersects the cube.*

We can solve the whole problem by collecting the solutions from each virtual cube that contains any particle(s). So we describe below how to obtain solution for a given cube (the seed), continued after the 3 steps of the initial setup.

We take the advantage that an interior point is provided with each element (say, for a high order element, pick the center node).

4) for a given seed cube, loop over the level of its neighbors until a cube is found to contain the center node (or pseudo node) some element (fig. 9);

The element found can be used as the starting one for local inclusion test with the particles contained in the seed cube. If necessary, one can use element face-searing relation to include more elements for local inclusion testing until solution is found.

At this stage, if one does not want to take the task of finding intersection between a curved element and a virtual cube, the following steps can be ignored. The reader can jump to section (2.8) for description of local inclusion tests.

Otherwise, the following steps can be performed to obtain elements intersecting the seed that contains some particles.

5) perform a directional walk on the boundary of the element found to find the cubes intersecting the element boundary (fig. 10);

6) use cube-wall sharing to identify cubes interior to the element, if any (this can be seen as an inverse process of 4);

7a) if the seed cube is interior, the tracer particle location problem is solved for the seed cube, and

7b) if the seed is found to be intersecting the element boundary, use element wall-sharing to identify the next element intersecting the seed until the seed is completely covered by the elements found;

8) then do a local inclusion test for each particles contained in the seed cube against the elements that cover the seed to find the owner element of each particle.

With solutions obtained for each seed (a virtual cube that contains any tracer particles), a loop over the seeds shall give us the solution to the whole problem. Before we go further, one has to be able to find the intersection between an element and a virtual cube. An effective way is to perform a directional walking on the boundary[1] (fig. 10).

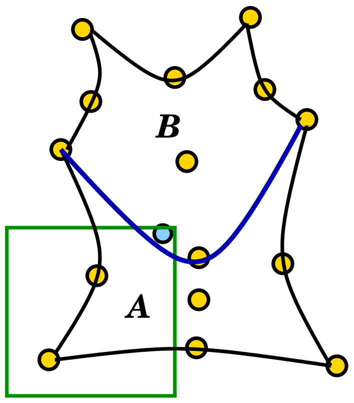

Figure 7: *A virtual cube (green square) is found to contain some pseudo nodes from element A. A particle (light blue dot) contained in the virtual cube is also contained in element B but no pseudo nodes from element B falls into the cube. In this case the particle is not inside A. However, starting from A and looping over neighbors of A with local inclusion tests would tell the particle is inside element B. New neighbor elements can always be added and the operation shall stop likely before all elements that intersect the cube are found with shared element-face (blue).*

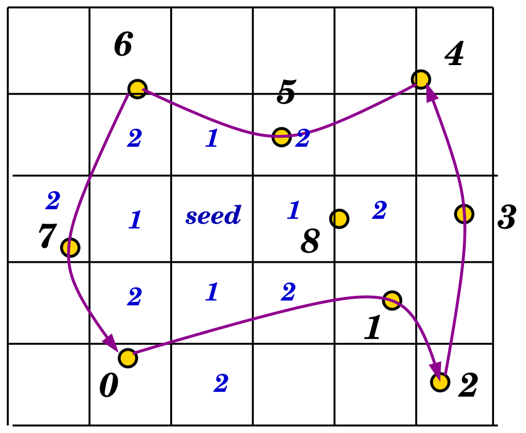

Figure 8: *Start from a seed cube, loop over level of neighbors using the wall-sharing relation to locate a neighbor cube that intersects a given element (by finding the virtual cube that contains the center node, say pseudo nodes may help also).*

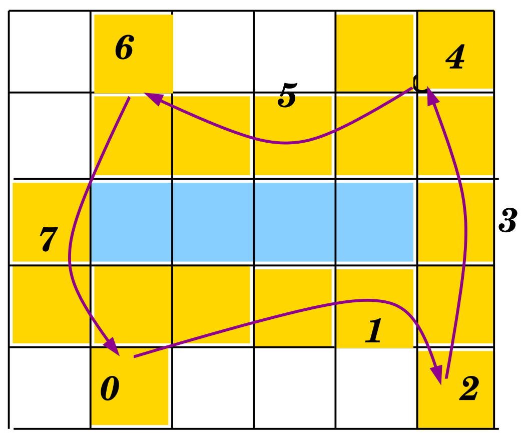

Figure 9: *A directional walk on the boundary of a curved element starts form node* 0. *A yellow cube contains some portion of the element boundary. Note the four light blue cubes completely contained in the curved element.*

2.6 A DIRECTIONAL WALK ON ELEMENT BOUNDARY

Since the sizes of elements in a high-order element system may vary significantly, some elements may be completely contained in a virtual cube (the element colored golden in fig. 3), or completely containing several virtual cubes (the green cube is contained in a curved-element in fig. 3). Otherwise, each element must be intersected by walls of the virtual cube complex.

With a loop over all nodes of the element system, each node shall find the ID of the virtual cube that contain the node by simply taking the integer parts of the physical coordinate of the node. A node at (x, y) is contained by the cube (i, j) with $i = (int)x, j = (int)y$. (fig. 3) By doing this, any element completely contained in a virtual cube will not be missed.

Now we are going to do a directional walk to find for each cell those virtual cubes intersecting the given cell. We demonstrate this operation in 2D first (fig. 10). It is assumed that the boundary can be expressed with $x_B = x(s), y_B = y(s)$ with s a parameter *strictly increasing* with the *arc-length* of boundary. For example, with a $Q2$ element, s can be linear to the logical parameter on each side.

The walk starts at node 0 (say, see fig. 10), the cell ID that contains this node is determined by the integer parts of the physical coordinates of the node. The

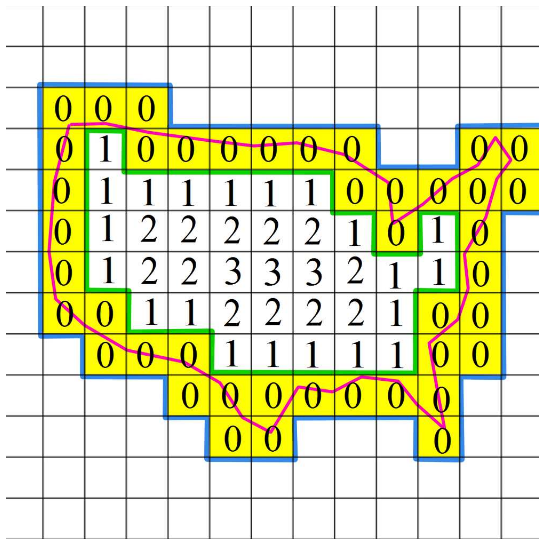
Figure 10: *Identify interior cubes with levels of wall sharing relationship with known interior walls of cubes. The integers mark the logical distances from a interior cube to boundary.*

walk path is the boundary of the curved cell. Once the walker meets a wall of the virtual cubes, he is going to enter the cube that share this wall with the cube he just walked through. When the starting node (0) is revisited, the walk is complete and all the cubes have been walked through are painted *yellow*. Note that the 4 light blue cubes are not walked in ever. They are *interior* to the element but how do determine this mathematically is a question and we will give the answer right away.

Once a yellow cube is being walked through, effectively the intersection of the element with the cube is determined because the points where the element boundary intersect the walls of a cube, are just vertices of the intersections between the cube and the element. A complete cube wall on the left of the walking path will be an *interior* wall. The collection of interior cell walls form the inner boundary of the yellow stripe of cubes that the walker has passed through. A cube that has not been walked through but share an *interior* wall with a yellow cube must be an interior cube to the given curved element.

If there are more interior cubes left, one finds them by continuously looking for cubes share walls with known interior cubes until everyone of them is found. This

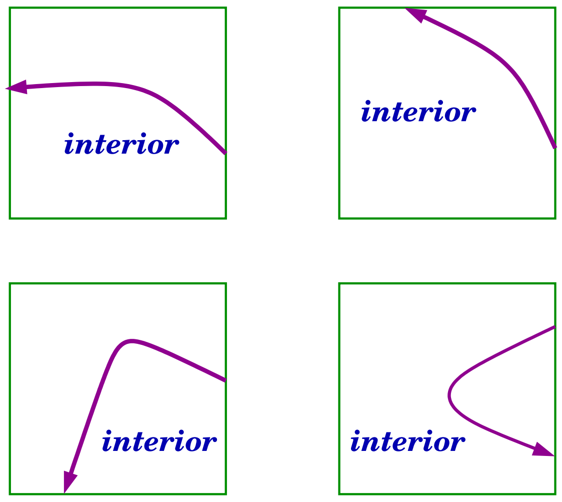

Figure 11: *Possible paths for walking in a virtual cube. With the walk in the arc-length increased direction, interior is by definition on the left of a path.*

is demonstrated in figure 11.

The case of the walker enters and exits a boundary cube on the same wall shall not be missed because if so, the arc-length becomes discontinuous and continuity of the path is broken.

In the three-dimensional case, similar things can be done. One starts from a node, and the cube contains this node is known to be identified by the integer parts of the physical coordinates of the node. Once the intersection of the element boundary with this cube is computed (say, with a directional walk described in [1]), the cube walls that intersect the element boundary are identified and one enters the cubes that share walls with the current cube (ignores walls that has no intersection with the boundary of the element). By keeping propagating the element boundary-cube intersections with the wall-sharing relationship, all cubes that contains a piece of the element boundary shall be identified. Not a single mix cube will be uncounted because otherwise, continuity of the boundary will be broken.

The directional walking in this situation involves intersection of a curved surface $f(x, y, z) = 0$ and a wall of the cubical complex $x = integer$ (or $y = integer$, or $z = integer$). In the most complex three-dimensional case, the intersection is a two-dimension curve $g(\xi, \eta) = 0$ with ξ, η a selected pair from x, y, and z. We will assume a 2D curve in this problem can be handled algebraically (or numerically).

2.7 IDENTIFY FOR EACH TRACER PARTICLE ITS HOST ELEMENT

We loop over the tracer particles, for each particle, find the virtual cube contains it (with taking integer parts of the physical coordinates of the particle). Then we check the list of elements intersecting this cube, and try to determine which element owns the particle. This is not as hard as imagined, we summarize the conditions here that

a) if the cube is intersected by a single element (like the blue ones in fig. 6), the particle is owned by the said element

b) if the cube is intersected by two elements A and B, one links this particle with a point that is in B on the wall on the intersection of B and the cube (this is computed at the stage of directional walking), with a line-segment. Then the number of intersections of this line-segment with the interface between A and B (this interface is known too from the directional walking) determines which element the particle is contained in. If the number is odd, the particle is in A, if even, the particle is in B.

c) if the cube is intersected by more than two elements, the bottom line is local inclusion tests can be done against each element from the local element list. Using a ray originated at the tracer particle to intersect each face of an element shall give the information if the particle is contained. If the number of intersections is *odd*, the particle is *exterior* to this element, if *even* the particle is *interior* to the element.

2.8 LOCAL INCISION TEST AND ALGEBRA FOR INTERSECTION

We demonstrate the algebraic approach to solve the problem of intersecting an element and a cube (for a directional walking), and local point inclusion test against a curved element to find tracer particle locations once the list of elements associated with the cube that contains a given tracer particle.

It is assumed that each element face is represented by a quadratic mapping to a straight line-segment, that a face with nodes A, B, and C is the mapping from $s = -1.0$, $s = 0.0$ and $s = 1.0$ in logical space with

$$x = \frac{1}{2}(x_A + x_C - 2x_B)s^2 + \frac{1}{2}(x_A - x_C)s + x_B, \qquad (1)$$

$$y = \frac{1}{2}(y_A + y_C - 2y_B)s^2 + \frac{1}{2}(y_A - y_C)s + y_B.$$

s is the logical parameter, its range is $[-1, 1]$. The direction of walk performed earlier is always in the s-increased direction on each face (the arc-length of face is strictly increasing with s, see fig. 10).

2.8.1 Intersection of Element Boundary and the Walls of a Cube

Because we assume the problem is scaled with a constant length (the side of a cube), the problem because finding intersections of an element face expressed in (eq. 1) with a line segment of length 1, with either $x = I$, or $y = J$, where I and J are some integers determines by the cube ID of the cube that the walker enters. Taking the cube wall $x = I$, $J \leq y < J + 1$ for example, and letting

$$a_0 = \frac{1}{2}(x_A + x_C - 2x_B), \qquad b_0 = \frac{1}{2}(x_A - x_C), \qquad c_0 = x_B - I,$$

and

$$a_1 = \frac{1}{2}(y_A + y_C - 2y_B), \qquad b_1 = \frac{1}{2}(y_A - y_C), \qquad c_1 = y_B,$$

the solution to $a_0 s^2 + b_0 s + c_0 = 0$ is straightforward with two roots that (assume a_0 is not zero so we are indeed dealing with a quadratic)

$$s_+ = -\frac{b_0 + \sqrt{b_0^2 - 4a_0c_0}}{2a_0},$$

and

$$s_- = -\frac{b_0 - \sqrt{b_0^2 - 4a_0c_0}}{2a_0}.$$

Then the two intersections are

$$x_+ = I, y_+ = a_1 s_+^2 + b_1 s_+ + c_1,$$

and

$$x_- = I, y_- = a_1 s_-^2 + b_1 s_+ + c_1.$$

We assume the element face ABC indeed intersect the given cube-wall (this is justified because an element is assumed not self-intersecting, a walker has to exit a cube from some wall once he has entered the cube). There can be only two cases that 1) there is one intersection; and 2) there are two intersections.

Case 1) is simple and it means a walker shall exit from the intersection (we do not check the entrance because it is the exit of the last cube the walker has entered) into another cube. In this case, either y_- is between J and $J+1$ with s_- residing in $[-1, +1)$ (means the intersection is *on* the face), or y_+ is between J and $J+1$ with s_+ residing in $[-1, +1)$.

Case 2) means both y_- and y_+ are between J and $J+1$ and both s_- and s_+ are contained in $[-1, +1)$. Two situations possible that a) the walker exits from the wall he entered in, back to the last cube he visited, and b), the walker exits this wall to the neighbor cube that shares this wall and comes back right away from another point on the same wall.

It is crucial that the walker marks each entrance and exit with the mileage (arc-length). The intersections shall be properly ordered this way, no confusion as long as the he does not go back (this is guaranteed by the assumption that the arc-length is a strictly increasing function of the logical parameter s on each element face).

2.8.2 Local Inclusion Tests with a Curved Element

We do a local inclusion test for a given tracer particle against a curved element by intersecting a ray starting from the point, and count the number of intersection with the faces of the element. If the number is *odd*, the point is inside the element, if *even*, it is outside.

Intersecting a ray defined by $x = x^* + r\ell_x, y = y^* + r\ell_y$ (where (x^*, y^*) is the origin of the ray (the location of a given point (tracer particle), and ℓ_x, ℓ_y are the unit directional vector of the ray) with an element face expressed by

$$x = a_0 s^2 + b_0 s + c_0, y = a_1 s^2 + b_1 s + c_1,$$

requires the solution of

$$r\ell_x = a_0 s^2 + b_0 s + c_0 - x^*, r\ell_y = a_1 s^2 + b_1 s + c_1 - y^*.$$

Because the direction of the ray can be arbitrary, we may take $\ell_x = 0, \ell_y = 1$ (say) and find the *two* intersections

$$s_+ = -\frac{b_0 + \sqrt{b_0^2 - 4a_0(c_0 - x^*)}}{2a_0}, r_+ = a_1 s_+^2 + b_1 s_+ + c_1 - y^*,$$

and

$$s_- = -\frac{b_0 - \sqrt{b_0^2 - 4a_0(c_0 - y^*)}}{2a_0}, r_- = a_1 s_-^2 + b_1 s_- + c_1 - y^*.$$

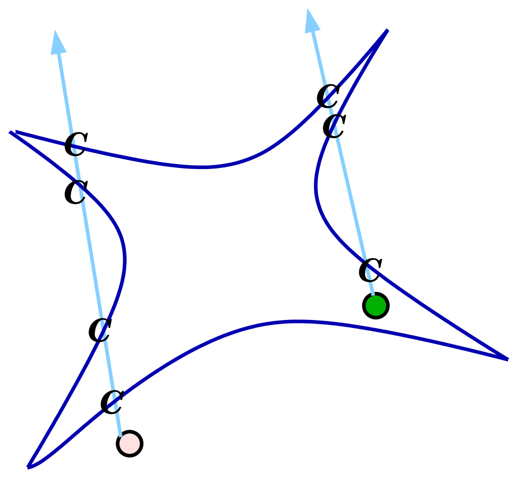

Figure 12: *Local inclusion tests with the algebra describes above. A ray originated from the pink particle intersected the boundary of the element 4 times so in exterior. A ray started from the green particle intersected the element boundary 3 times, so in interior. Intersections (crossing points) are marked with letter C.*

If $-1 \leq s_+ < +1$ and $r_+ > 0$, increase the count of intersections by 1, if $-1 \leq s_- < +1$ and $r_- > 0$, also increase the count by 1, a double root can be counted twice (or ignored).

Loop over all the four faces of this element and add the counts, one would figure out if a point (tracer particle) is in a given element. Because the list of local elements associated with a give cube (the one contains the given particle) completely cover the cube, then after every local element is checked, the owner element of the given particle is found out. (fig. 13).

For higher order elements, an explicit expression of the solution may not be available. An iterative method may have to be used. However the problem of intersecting a ray with a surface is essentially an one-dimensional search on the ray starting from the origin. This feature would provide simplification of the algebra necessary for finding the solution.

3 CONCLUSION

With a characteristic length to scale the problem, a virtual cubical complex is established to cover the computational domain. The problem of finding tracer particle locations in a curved element system is then decomposed to a collection of problems to identify a list of elements that cover a given non-empty virtual cube (the *seed*) that contains some particle(s). With the wall-sharing relations between neighbor virtual cubes and neighbor elements, in addition to a directional walk on the boundary of an element to find intersections with local cubes, a list of elements that cover a given virtual cube is built. (Putting pseudo nodes on element boundary or interior of an element may help to avoid some unnecessary intersections). Local inclusion tests then can be performed to find the solution for any given seed. A loop over all the seed cubes shall give the whole solution of the problem.

The proposed method is local and involves only elements that intersect non-empty virtual cubes so is efficient. It would also work for an arbitrary planar mesh similarly.

ACKNOWLEDGMENTS

The author appreciates discussions with Rob Rieben, Tzanio Kolev, and Veselin Dobrev at LLNL regarding this problem. Their technical knowledge and expertise provided the author with interest and guidance.

REFERENCE

[1]. Yao, Jin, Directional Walking: A Simple Way to Intersect Arbitrary Geometries, *LLNL-JRNL-400055*, 2007.

[2]. Yao, Jin, An Efficient Inclusion Test for a Massive Point Distribution, *LLNL-TR-636668*, 2013.

[3]. Rieben, Robert, *personal communication*, 2013

[4]. Kolev, Tzanio, *personal communication*, 2013

[5]. Dobrev, Veselin, *personal communication*, 2013